

AI DRIVEN STAFF ALLOCATION AND SCHEDULING SYSTEM

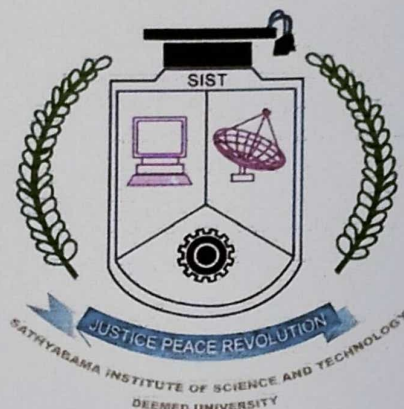
Submitted in partial fulfillment of the required for the award of

Masters of Computer Science

by

PAVITHRA A

REGISTER NO (42734027)



DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF SCIENCE AND HUMANITIES

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600 119

APRIL 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC I Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of
PAVITHRA A (42734027) who carried out the project entitled "**AI-Driven
Staff Allocation and Scheduling System**" under our supervision from January
2024 to April 2024.

INTERNAL GUIDE

Mrs. Jesintha J M.Sc., Assistant Professor

DEAN AND HEAD OF THE DEPARTMENT

Dr. Rekha Chakaravarthi M.E., Ph.D.,

Submitted for Viva voce examination held on

20/4/2024

Internal Examiner

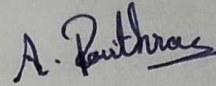
External Examiner

DECLARATION

I, **PAVITHRA A (42734027)** hereby declare that the Project Report entitled **"AI-Driven Staff Allocation and Scheduling System"** done by me under the guidance of **Mrs.Jesintha J M.Sc., Assistant Professor**, Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600 119 is submitted in partial fulfilment of the requirements for the award of Master of Science degree in Computer Science.

DATE: 20/4/2024

PLACE: CHENNAI



PAVITHRA A

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.REKHA CHAKRAVARTHI M.E. Ph.D., Dean of Arts and Science** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my project guide **Mrs. Jesintha J M.Sc., Assistant Professor**, Department of Computer Science, for her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff member of the Department of Computer Science who were helpful in many ways for the completion of the project.

PAVITHRA A

ABSTRACT

Efficient allocation of resources and time management are critical in the operation of any educational institution, particularly in colleges where numerous courses and activities need to be coordinated. Traditional manual methods of creating timetables are time-consuming and prone to errors. In this project, the system propose a staff-driven automatic college timetable scheduling system aimed at streamlining the scheduling process while accommodating the preferences and availability of teaching staff. The system employs a constraint-based approach, integrating staff availability, course requirements, classroom capacities, and other constraints to generate optimal timetables. Various scheduling algorithms, including Constraint Satisfaction Programming (CSP) and Genetic Algorithms (GA), are explored to find the most suitable solution. The implementation of the system involves data gathering, constraint definition, modeling, software development, testing, and deployment. Through the deployment of the system, colleges can expect improved efficiency, reduced administrative burden, and enhanced satisfaction among staff and students. Future enhancements and optimizations, such as machine learning techniques for adaptive scheduling and integration with student preferences, are also discussed. Overall, the proposed system presents a practical solution to the complex task of college timetable scheduling, leveraging automation to streamline operations and enhance organizational effectiveness.

Chapter No	TITLE	Page No
	ABSTRACT	i
1	INTRODUCTION	1
1.1	Project Description	1
2.	SYSTEM ANALYSIS	2
2.1	Existing System	2
2.2	Proposed System	3
3	LITERATURE SURVEY	5
4	SYSTEM ARCHITECTURE	10
5	SYSTEM DESIGN	13
5.1	UML Diagrams	13
5.1	Use Case Diagram	13
5.2	Sequence Diagram	14
5.3	Activity Diagram	15
5.4	Package Diagram	16
5.5	Component Diagram	17
6	MODULES	19
6.1	Entity Access	19
6.2	Staff Details	19
6.3	RNN Allocation	20
6.4	Scheduling Verification	20
6.5	Time Table Generator	21
7	REQUIREMENT SPECIFICATION	22
7.1	Introduction	22
7.2	Hardware Requirements	22
7.3	Software Specification	23

7.4	Genetic Algorithm	23
7.5	General working of Genetic Algorithm	23
7.6	Working of Genetic Algorithm in Time Table Generator	25
8	TESTING AND VALIDATION	27
8.1	Unit Testing	27
8.2	Integration Testing	27
8.3	User Acceptance Testing	28
8.4	Performance Testing	28
9	CODING AND SAMPLE OUTPUT	29
9.1	Source Code	29
9.2	Samples Screenshots	35
10	CONCLUSION AND FUTURE WORK	38
10.1	Conclusion	38
10.2	Future Enhancement	38

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO
4.1	System Architecture	11
5.1	Use case diagram	14
5.2	Sequence Diagram	15
5.3	Activity Diagram	16
5.4	Package Detection	17
5.5	Component diagram	18
7.1	Flow Chart for Genetic Algorithm	24
9.1	Home Page	35
9.2	Adding Staff Details	35
9.3	Adding Course Details	36
9.4	Adding Department	36
9.5	Adding Section	37
9.6	Final Time Table	37

LIST OF TABLES

TABLE NO	NAME OF THE TABLE	PAGE NO
3.1	Summary of Literature Survey	9

CHAPTER 1

INTRODUCTION

Efficient scheduling of college timetables is essential for ensuring smooth academic operations, maximizing resource utilization, and providing an optimal learning environment for students. However, the process of creating timetables manually can be arduous, prone to errors, and often fails to accommodate the preferences and constraints of teaching staff. Automated timetable scheduling systems offer enhanced flexibility and adaptability compared to manual scheduling processes. As colleges expand their course offerings and adapt to changing academic needs, the complexity of timetable scheduling increases exponentially.

In response to these challenges, automated timetable scheduling systems have emerged as promising solutions to streamline the scheduling process and improve overall efficiency. These systems leverage computational algorithms and optimization techniques to generate timetables that meet various constraints and preferences while minimizing conflicts.

1.1 PROJECT DESCRIPTION

In this project, a staff-driven automatic college timetable scheduling system designed to address the specific needs of academic institutions is developed. Unlike traditional approaches that focus solely on course requirements and resource availability, our system prioritizes the preferences and availability of teaching staff, recognizing their pivotal role in the delivery of quality education.

The system utilizes a constraint-based approach, integrating staff availability, course requirements, classroom capacities, and other constraints to generate optimal timetables. We explore various scheduling algorithms, including Constraint Satisfaction Programming (CSP), Genetic Algorithms (GA), and hybrid approaches, to find solutions that balance conflicting objectives and constraints effectively. User-friendly interfaces and intuitive controls empower staff members to actively participate in the scheduling process, fostering a sense of ownership and engagement.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In many educational institutions, the process of creating college timetables is primarily manual, relying on spreadsheets, paper-based forms, and ad-hoc methods. While some colleges may use basic scheduling software, these tools often lack the flexibility and intelligence needed to handle the complexities of timetable generation effectively. As a result, administrators and staff face numerous challenges in creating timetables that balance course requirements, staff preferences, and resource constraints.

Manual timetable scheduling is labor-intensive and time-consuming, requiring administrators to coordinate with department heads, faculty members, and support staff to gather availability data and resolve scheduling conflicts. Moreover, the static nature of manual schedules makes it difficult to accommodate last-minute changes, such as staff absences or room conflicts, without disrupting the entire timetable.

Furthermore, manual scheduling processes are susceptible to errors, leading to inaccuracies in timetables and potential disruptions to the academic calendar. Additionally, the lack of visibility and transparency in manual scheduling systems can hinder collaboration and communication among stakeholders, resulting in inefficiencies and frustration.

While some colleges may use basic scheduling software, these tools often lack the flexibility and intelligence needed to handle the complexities of timetable generation effectively. They may offer limited customization options and lack features to optimize schedules based on staff preferences, course requirements, and other constraints.

Overall, the existing system of manual college timetable scheduling is fraught with inefficiencies, errors, and challenges that hinder academic operations and compromise the quality of education. As colleges continue to grow and adapt to changing needs, there is a pressing need for more advanced and automated scheduling solutions to streamline operations, improve efficiency, and enhance the overall academic experience for students and staff alike.

2.2 PROPOSED SYSTEM

The proposed system aims to revolutionize the college timetable scheduling process by introducing automation while prioritizing the preferences and constraints of teaching staff. Here's an overview of the key components and features:

Staff Preferences Integration: The system will allow teaching staff to input their availability preferences, preferred courses to teach, and preferred time slots. This information will be used as constraints in the scheduling algorithm to ensure that staff preferences are considered during timetable generation.

Course Requirements and Constraints: Course requirements such as duration, prerequisites, and preferred timings will be integrated into the system. Additionally, constraints such as classroom capacities, room availability, and institutional policies will be taken into account to generate feasible timetables.

Constraint-Based Scheduling Algorithm: The system will employ a constraint-based scheduling algorithm that considers staff preferences, course requirements, and institutional constraints to generate optimal timetables. Techniques such as Constraint Satisfaction Programming (CSP) or Genetic Algorithms (GA) will be explored to find efficient solutions.

User-Friendly Interface: The system will feature a user-friendly interface accessible to administrators, department heads, and teaching staff. Administrators can input constraints, view scheduling progress, and approve final timetables. Teaching staff can input their preferences and view their assigned schedules.

Real-Time Updates and Notifications: The system will provide real-time updates and notifications to users regarding scheduling progress, changes, and conflicts. This ensures transparency and allows for timely adjustments to schedules as needed.

Flexibility and Adaptability: The system will be designed to accommodate last-minute changes, such as staff absences or room conflicts, without disrupting the entire timetable. It will offer flexibility to re-optimize schedules based on updated constraints and preferences.

Integration with Existing Systems: The system will seamlessly integrate with existing college systems such as student information systems (SIS) and learning management systems (LMS). This integration ensures data consistency and facilitates information sharing across different platforms.

Testing and Validation: Prior to deployment, the system will undergo rigorous testing to ensure accuracy, reliability, and performance. Testing will involve simulating various scenarios and evaluating scheduling outcomes against predefined criteria.

Training and Support: Comprehensive training and support will be provided to users to ensure effective utilization of the system. Training sessions will cover system functionality, scheduling best practices, and troubleshooting procedures.

Therefore, the proposed system offers a comprehensive solution to automate the college timetable scheduling process while considering the preferences and constraints of teaching staff. By streamlining operations and enhancing efficiency, the system aims to improve the overall academic experience for staff and students alike.

CHAPTER 3

LITERATURE SURVEY

3.1 PAPER TITLE: The Examination Timetabling Problem Based on Expert System: A Case Study in Malaysia.

AUTHOR: Cawangan Johor, Kampus Pasir Gudang;

YEAR OF PUBLISHING: March 2018 – June 2018 and September 2018 – January 2019;

JOURNAL NAME: IEEE 10th Control System and System Graduate Research Colloquium (ICSGRC)

The Examination Management System (eMaS) is specifically developed for managing and generating all aspects regarding final examination matters such as the timetable for students and invigilators, reports on the occupancy of examination rooms and other examination documents that correspond with the Department of Academic Affairs (BHEA). eMaS provides valuable features that grant precise and fast solution for improving the existing examination scheduling system. All constraints raised such as invigilators' special requests, and specific time slots for certain exams will be scrutinized and automatically generate instantly. The generated timetable can be accessed online easily. eMaS system is developed by using the Visual Basic (VB), Artificial Intelligence (AI) and Graphical User Interface (GUI). To confirm with the competency of the developed eMaS, University Teknologi MARA (UiTM) Cawangan Johor, Kampus Pasir Gudang (UiTMCJKPG) has been chosen as to pilot the project. The system has been successfully deployed in automatically generating the final examination timetable for the semester sessions of March 2018 – June 2018 and September 2018 – January 2019 respectively. Reviews obtained from the piloted project proved that eMaS offered a precise time saving solution on the scheduling management while improving the quality and reliability of the system.

3.2 PAPER TITLE: Intergrated Cyclic Train Timetabling and Rolling Stock Circulation Planning For a Suburban Railway Line.

AUTHOR: Yuhui Wang, Wenzhang Huang, Miaoyu Ju, Ai Ren;

YEAR OF PUBLISHING: Septmeber 2023

JOURNAL NAME: IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)

With the steady development of China's economy and the progress of the urbanization process, the suburban railway has become an important role in transportation for passengers to travel locally. Reasonable timetable can not only improve operational efficiency, but also improve passenger satisfaction. In this paper, an integrated cyclic train timetabling and rolling stock circulation planning model considering passenger demand is established for a Chinese suburban railway line based on periodic event scheduling problem (PESP). Specifically, this model considers the possibility of overtaking and flexible connection of rolling stocks, i.e., a set of rolling stocks may perform different type of train services in different periods. A lagrangian relaxation algorithm is developed to shorten the computation time. The effectiveness of the model is demonstrated by several numerical experiments based on the available data of Jingxiong R1 line.

3.3 PAPER TITLE: Reinforcement Learning for Scalable Train Time Table Rescheduling with Graph Representation.

AUTHOR: Yue Peng, Yachu Jin, Xuewu Dai, Zhenhua Feng;

YEAR OF PUBLISHING: January 2024

JOURNAL NAME: IEEE Transactions on Intelligent Transportation Systems PP(99)

Train timetable rescheduling (TTR) aims to promptly restore the original operation of trains after unexpected disturbances or disruptions. Currently, this work is still done manually by train dispatchers, which is challenging to maintain performance under various problem instances. To mitigate this issue, this study proposes a reinforcement learning-based approach to TTR, which makes the following contributions compared to existing work. First, we design a simple directed graph to represent the TTR problem, enabling the automatic extraction of informative states through graph neural networks. Second, we reformulate the construction process of TTR's solution, not only decoupling the decision model from the problem size but also ensuring the

generated scheme's feasibility. Third, we design a learning curriculum for our model to handle the scenarios with different levels of delay. Finally, a simple local search method is proposed to assist the learned decision model, which can significantly improve solution quality with little additional computation cost, further enhancing the practical value of our method. Extensive experimental results demonstrate the effectiveness of our method. The learned decision model can achieve better performance for various problems with varying degrees of train delay and different scales when compared to handcrafted rules and state-of-the-art solvers.

3.4 PAPER TITLE: Intelligent Automation & Soft Computing; Genetic Algorithms vs Tabu Search in Time Table Scheduling.

AUTHOR: Abayomi Alli Adebayo, Sanjay Misra, Olusola Oluwakemi Abayomi Alli;

YEAR OF PUBLISHING: January 2020

JOURNAL NAME: IEEE Third International Conference on knowledge-Based Intelligent Information Engineering Systems.

In a timetabling problem, exam subjects must be slotted to certain times that satisfy several of constraints. A web-based expert system, called class schedule planner (CSP). CSP encapsulates class-scheduling knowledge and gives intelligent scheduling advises to students. CSP has a set of Web forms to collect inputs from the users and then translates the request into facts. It uses JESS (Java expert system shell) as a tool to process the facts and rules to generate feasible schedules for the students. Integration of data from various sources into JESS facts and rules is done by CSP controller and XML translator. The translated result from JESS is integrated with the Web presentation using JSP (Java server page). The unique technical contribution of this work is that unlike most other expert systems that require static expert knowledge this expert system allows dynamic management of knowledge in real time using Web interface. They are NP-completeness problems, which usually lead to satisfactory but sub-optimal solutions. This paper investigates and compares Genetic Algorithm and Tabu Search approaches to solve these kinds of problem. The experiment shows that TS approach can produce better timetables than those of GA approach can. Further, the search time spent in TS is less than that of GA. However, GA can produce several different near optimal solutions simultaneously.

3.5 PAPER TITLE: Auto-generate scheduling system based on expert system.

AUTHOR NAME: Nur Iqtiyani Ilham; E. H. Mat Saat; N. H. Abdul Rahman; Farah Yasmin
Abdul Rahman; Nurhani Kasuan.

YEAR OF PUBLISHING: November 2017

JOURNAL NAME: IEEE International Conference on Control System, Computing and Engineering (ICCSCE)

This paper proposed a technique for smart autogenerate scheduling system specifically for the educational sector. In constructing a precise and high efficient timetable there are constraints that needs to be conceded i.e. availability of class rooms, students, lecturers, courses, time slots etc. These are the tedious elements that contribute to the challenges in producing the same. Considering Faculty of Electrical Engineering (FKE) University Teknologi MARA (UiTM), Pasir Gudang campus as a piloted project, the proposed Auto-Generated Scheduling System (AGSS) is expected to overwhelm these problems. AGSS will provide the accessibility to the timetable committee to arrange the detail by simply loaded the information i.e. numbers of lecturer, list of class room, courses and loading detail (ATS) into the developed algorithm Artificial Intelligence (AI) expert system. Xampp and Visual basic is used in developing the timetable database and Graphical User Interface (GUI) for timetable system respectively. Based on the loaded information, the system will generate the class timetable automatically with individual user customizable setting. AGSS is adept to envisage the cost effective with fast and precise solution on the timetable management thus providing alternative solutions for timetable management while maintaining quality, reliability, and functionality.

In summary, by integrating innovative techniques, considering staff preferences, and addressing real-world challenges, researchers aim to develop more efficient and effective scheduling solutions to improve the academic experience for all stakeholders.

S.NO	PAPER TITLE	JOURNAL NAME	AUTHOR NAME	YEAR OF PUBLISHING
1.	The Examination Timetabling Problem Based on Expert System: A Case Study in Malaysia.	IEEE	Cawangan Johor, Kampus Pasir Gudang;	2019
2.	Intergrated Cyclic Train Timetabling and Rolling Stock Circulation Planning For a Suburban Railway Line.	IEEE 26th International Conference (ITSC);	Yuhui Wang, Wenzhang Huang, Miaoyu Ju, Ai Ren;	2023
3.	Reinforcement Learning for Scalable Train Time Table Rescheduling with Graph Representation.	IEEE Transactions on Intelligent Transportation Systems PP	Yue Peng, Yachu Jin, Xuewu Dai, Zhenhua Feng;	2024
4.	Intelligent Automation & Soft Computing; Genetic Alogrithms vs Tabu Search in Time Table Scheduling.	Third International Conference	Abayomi Alli Adebayo, Sanjay Misra, Olusola Oluwakemi Abayomi Alli;	2020
5.	Auto-generate scheduling system based on expert system.	IEEE	Nur Iqtiyani Ilham; E. H. Mat Saat; N. H. Abdul Rahman; Farah Yasmin Abdul Rahman; Nurhani Kasuan;	2017

Table 3.1 Summary of Literature Survey

CHAPTER 4

SYSTEM ARCHITECTURE

System architecture refers to the conceptual structure and organization of a complex system. It defines the components or modules of the system, their relationships, and how they interact to achieve the system's objectives. A well-designed system architecture provides a blueprint for implementing and managing the system effectively.

In software engineering, system architecture typically encompasses:

- **Components:** Identifying the high-level building blocks or modules that make up the system. These components encapsulate specific functionalities or services.
- **Interfaces:** Defining the interfaces between components, specifying how they communicate and interact with each other. Interfaces include methods, protocols, data formats, and communication channels.
- **Data Management:** Describing how data is structured, stored, processed, and accessed within the system. This includes databases, data flows, data models, and data synchronization mechanisms.
- **Deployment:** Planning how the system will be deployed in various environments, such as on-premises servers, cloud infrastructure, or distributed networks. Deployment considerations include scalability, availability, security, and resource allocation.
- **Concurrency and Communication:** Addressing how concurrent processes or threads interact and synchronize within the system. This includes communication patterns, synchronization mechanisms, and handling concurrency issues.
- **Scalability and Performance:** Ensuring that the system architecture can accommodate growth and handle increasing loads efficiently. This involves designing for horizontal and vertical scalability, optimizing performance-critical components, and minimizing bottlenecks.
- **Security:** Incorporating security measures to protect the system against unauthorized access, data breaches, and other security threats. Security considerations include authentication, authorization, encryption, and auditing.

- **Reliability and Fault Tolerance:** Designing the system to be robust and resilient in the face of failures or unexpected events. This may involve redundancy, fault tolerance mechanisms, error handling, and recovery strategies.
- **Maintainability and Extensibility:** Architecting the system in a way that facilitates ease of maintenance, updates, and future enhancements. This includes modular design, clear separation of concerns, and adherence to coding standards and best practices.

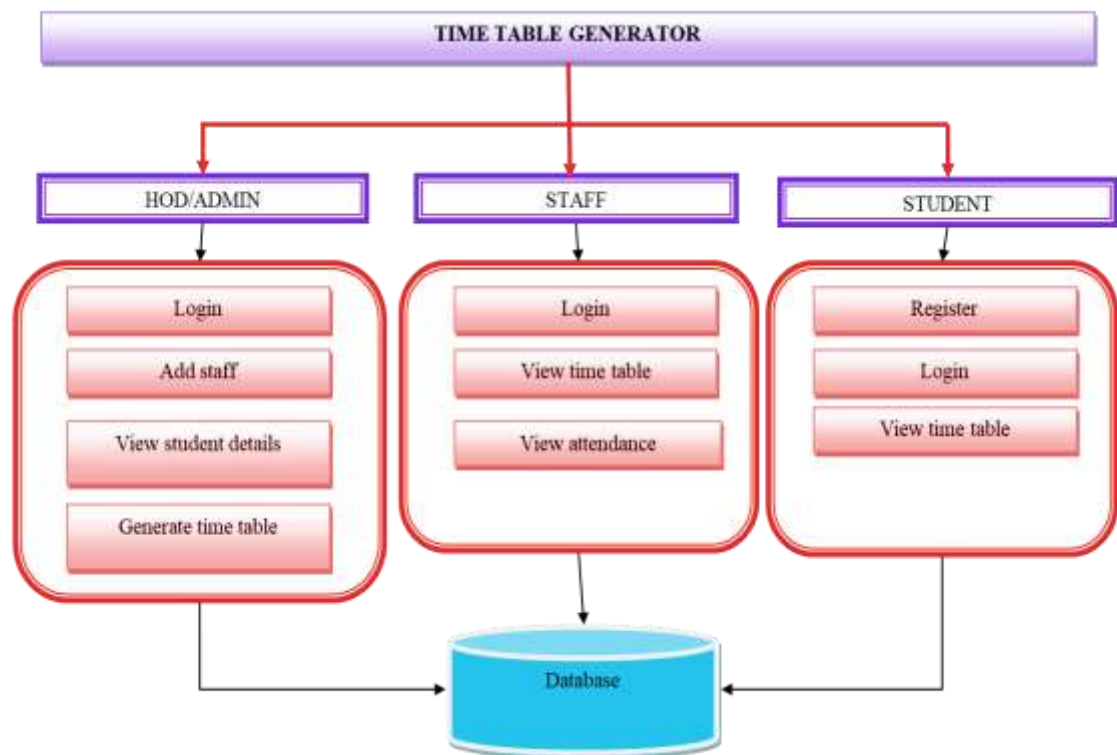


Fig 4.1 System Architecture for Time Table Generator

In the System Architecture Diagram:

1. Login and Registration

- Admins can securely log in to the system using their credentials.
- New users can register themselves by providing necessary details such as email, username, and password.

2. Add Staff

- After successful login and registration, Admins can add staff members and access to the system.
- Staff details, including email ID and user ID, are recorded for each staff member.

3. Department and Year Setup

- Admins can set up departments and specify the corresponding year or class details within the application.
- This step ensures proper organization and categorization of classes for timetable generation.

4. Timetable Generation

- The Admin has the authority to generate the timetable for classes based on the predefined department and year setup.
- The timetable generation process is automated, leveraging the system's scheduling algorithm to optimize class schedules and staff assignments.

5. Staff Login

- Staff members can log in using their email ID and password credentials.
- After login, staff members can view the timetable of classes allocated to them.

6. Student Registration

- Students need to register with their email ID and date of birth to access the timetable.
- Upon registration, students can log in at any time to view the timetable.

CHAPTER 5

SYSTEM DESIGN

A UML (Unified Modeling Language) diagram is a graphical representation of a system or process using standardized symbols and notation. UML diagrams are widely used in software engineering and other fields to visually depict various aspects of a system, such as its structure, behavior, interactions, and architecture.

There are several types of UML diagrams, each serving a different purpose and focusing on different aspects of the system being modeled. Some common types of UML diagrams include:

Unified Modeling Language (UML) diagram for the proposed automated college timetable scheduling system. Here's an example of a high-level UML class diagram for the system:

5.1 USE CASE DIAGRAM

Use case diagram illustrates the interactions between different actors (roles) and the system itself. It provides a high-level overview of the system's functionalities from the perspective of different users. Additional use cases can be added to capture more detailed interactions and functionalities.

In this Use Case Diagram:

College Timetable: Represents the main functionality of the system, which is generating the timetable.

Administrator: Can input constraints such as staff availability, room capacities, and course requirements. They also have the authority to approve the final timetable. Manages users (administrators, staff, etc.) and system settings.

Staff: Can input their preferences regarding teaching schedules and courses.

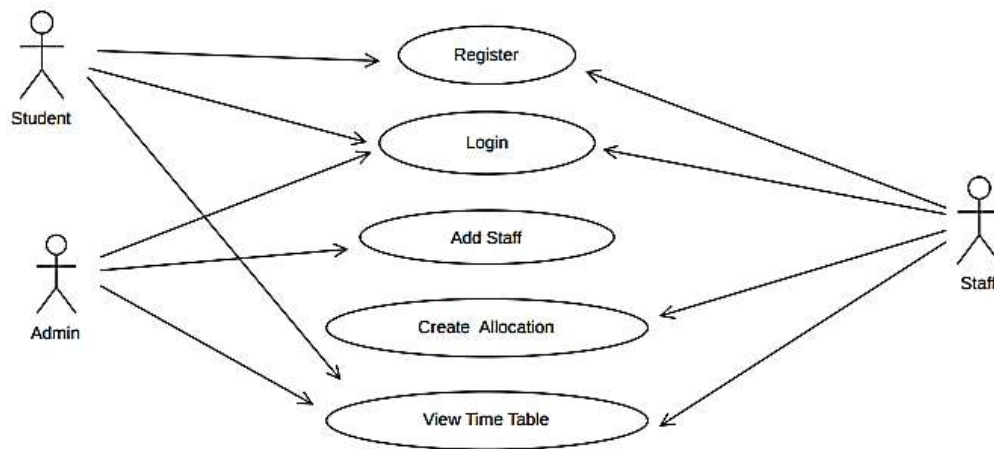


Fig 5.1 Use case diagram for admin, student and staff

5.2 SEQUENCE DIGRAM

Sequence Diagram or system sequence diagram (SSD) shows process interactions arranged in a time sequence. The diagram depicts the processes and objects involved and the sequence of messages exchanged as needed to carry out the functionality.

In this Sequence Diagram:

The Administrator interacts with the system by inputting constraints via the input Constraints() method. The College Timetable System receives the constraints and proceeds to generate a timetable using the generate Timetable() method. After generating the timetable, the system displays it to the Administrator via the display Generated Timetable() method. The Administrator approves the timetable using the approve Timetable() method. Upon approval, the system notifies the staff about the finalized timetable using the notify Staff() method.

This sequence diagram illustrates the sequential flow of interactions between the Administrator, the College Timetable System, and the Staff during the process of generating and approving a timetable.

Below is a simplified sequence diagram illustrating the interaction between an Administrator and the College Timetable System to generate and approve a timetable.

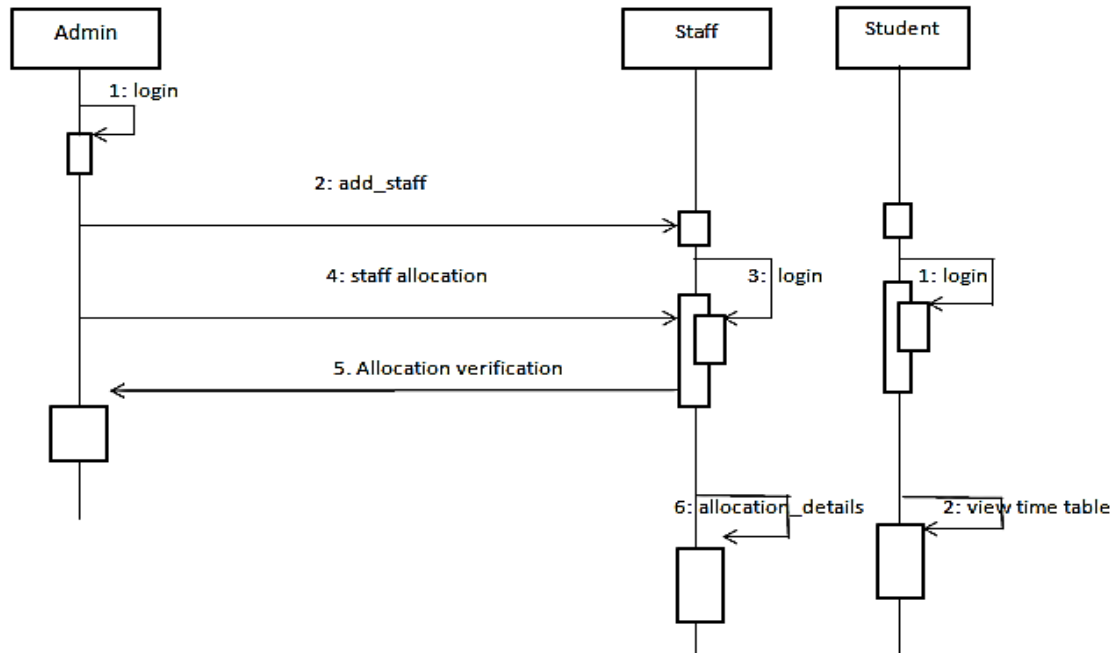


Fig 5.2 Sequence Diagram for admin, student, staff allocation and scheduling system

5.3 ACTIVITY DIAGRAM

Activity Diagrams are used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. It is a type of behavioral diagram and we can depict both sequential processing and concurrent processing of activities using an activity diagram. An activity diagram focuses on the condition of flow and the sequence.

In this Activity Diagram:

The Administrator inputs constraints into the system, such as staff availability, course requirements, and room capacities. The College Timetable System processes the constraints and generates a timetable based on the input. The Administrator reviews the generated timetable and approves it. The approved timetable is then made available for staff to view. This activity

diagram provides a high-level overview of the steps involved in generating and approving a college timetable within the automated system.

Below is the simplified Activity diagram illustrating the process of generating a college timetable within the automated system:

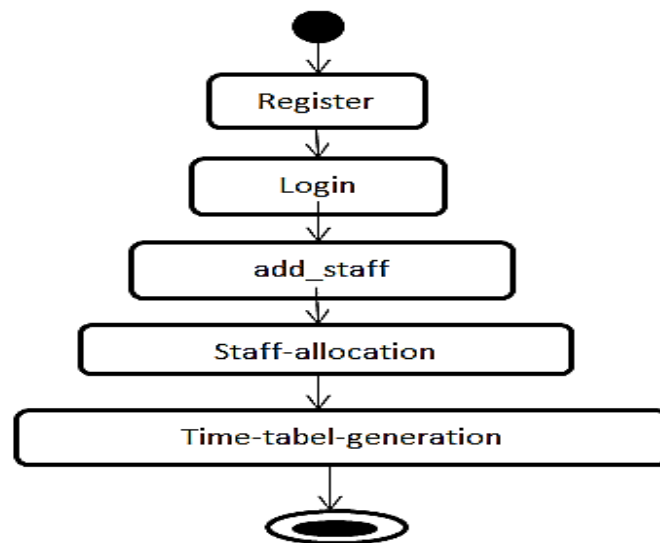


Fig5.3 Activity Diagram for Time Table Generation

5.4 PACKAGE DIGRAM

A package diagram is a type of Unified Modeling Language (UML) diagram mainly used to represent the organization and the structure of a system in the form of packages.

In this package diagram:

College Timetable: Represents the main package containing all the classes and components related to the college timetable scheduling system.

Administrator: Package containing classes related to administrators' functionalities.

Staff: Package containing classes related to staff functionalities.

System Admin: Package containing classes related to system administration functionalities.

Timetable Generator: Package containing classes related to timetable generation functionalities.

Below is a simplified package diagram representing the package structure of the automated college timetable scheduling system:

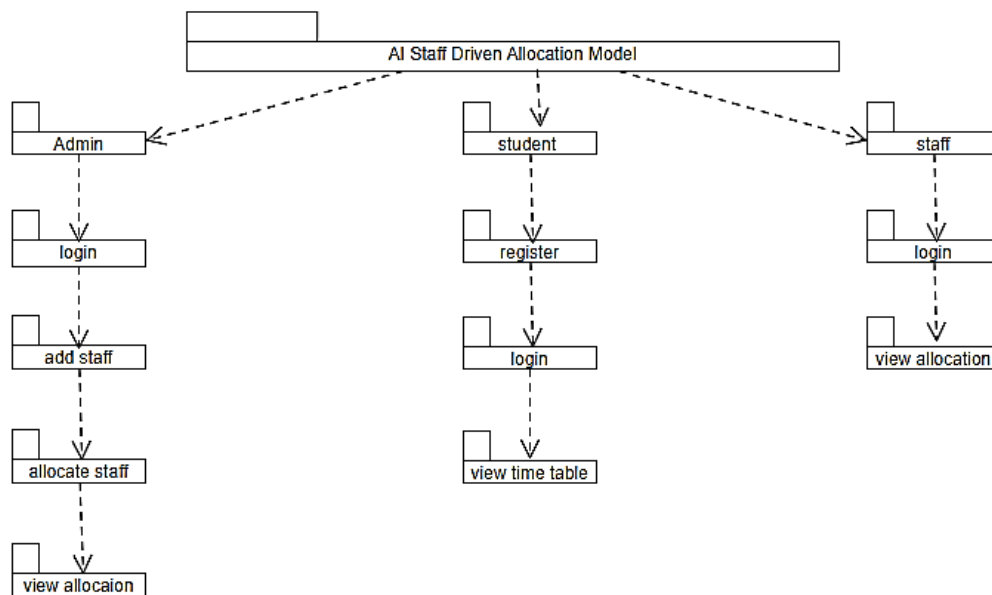


Fig 5.4 Package Detection for staff allocation and view time table

This package diagram provides a high-level overview of the package structure of the automated college timetable scheduling system, organizing classes and components into logical groupings based on their functionalities.

5.5 COMPONENT DIAGRAM

Component Diagrams are used to show code modules of a system in Unified Modeling Language (UML). They are generally used for modeling subsystems. It represents how each

and every component acts during execution and running of a system program. They are also used to show and represent structure and organization of all components.

In this Component Diagram:

College Timetable: Represents the core component responsible for managing the overall timetable generation and approval process. It interacts with the UI components to receive inputs and provide outputs.

Timetable Generator: This component is responsible for generating and optimizing timetables based on the inputs received from the UI components.

Staff Management: This component manages staff-related functionalities such as staff assignments and course allocations.

The arrows between components represent the flow of information and control between the components. This diagram provides an overview of the components and their interactions within the automated college timetable scheduling system.

Below is a simplified component diagram representing the components and their interactions in the automated college timetable scheduling system:

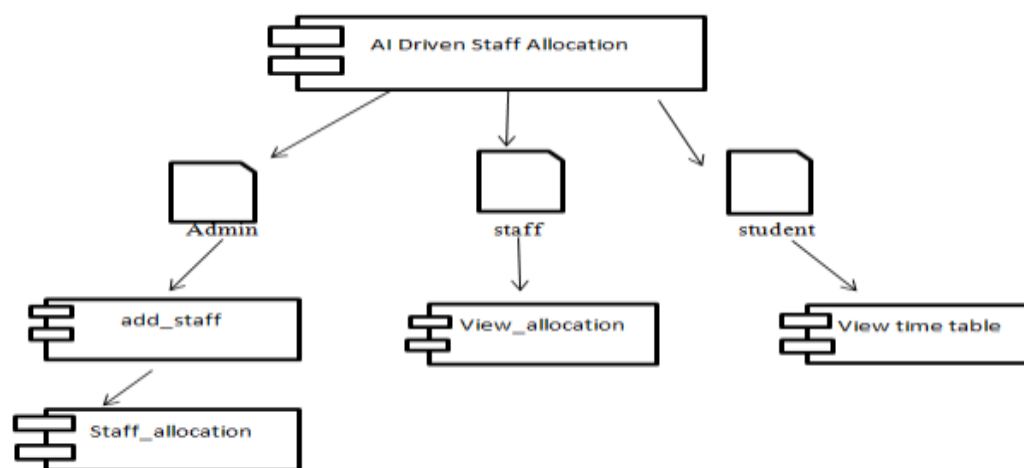


Fig 5.5 Component diagram for adding staff details and view allocation

CHAPTER 6

MODULES

6.1 ENTITY ACCESS

The Entity Access component serves as the backbone of authentication and authorization within the system. It encompasses functionalities essential for managing user accounts, permissions, and access control. Key features include:

- **User Registration:** Provides a seamless process for users to create new accounts by supplying necessary information such as username, email, and password.
- **Login/Logout:** Enables users to securely authenticate themselves into the system and logout when their session is complete.
- **Password Management:** Facilitates password reset, change, and recovery mechanisms to ensure account security.
- **Role-Based Access Control (RBAC):** Implements role-based permissions to regulate access to different parts of the system based on user roles (e.g., admin, staff, student).

6.2 STAFF DETAILS

The Staff Details centralizes the management of staff-related information, fostering efficient organization and retrieval of staff data. Notable functionalities include:

- **Profile Management:** Allows administrators to create, update, and delete staff profiles, including personal details, contact information, job roles, and qualifications.
- **Search and Retrieval:** Provides robust search capabilities to locate staff members based on various criteria, enhancing accessibility and usability.

6.3 RNN ALLOCATION

RNN (Recurrent Neural Network) allocation likely refers to using neural network-based algorithms to optimize resource allocation, specifically in the context of staff assignment to specific tasks or responsibilities within the timetable scheduling system.

- **Data Representation:** The first step involves representing the scheduling problem in a format suitable for input into the RNN model. This could include encoding staff availability, course requirements, room capacities, and other relevant constraints as input features.
- **Model Training:** The RNN model is trained using historical data on staff assignments, scheduling preferences, and other relevant information. The model learns to identify patterns and relationships between input variables and output assignments.
- **Optimization Objective:** The objective of the RNN allocation algorithm is to optimize resource allocation based on predefined criteria and constraints. This could include maximizing staff preferences, minimizing conflicts, and ensuring efficient resource utilization.
- **Integration with Scheduling System:** The optimized staff assignments generated by the RNN allocation algorithm are integrated into the timetable scheduling system, where they are used to generate the final timetable. This ensures that staff members are assigned to tasks or responsibilities in a manner that maximizes efficiency and meets scheduling requirements.

6.4 SCHEDULING VERIFICATION

The Scheduling Verification component ensures the integrity and validity of schedules by enforcing predefined criteria and constraints. It encompasses functionalities such as:

- **Conflict Detection:** Detects conflicts within schedules, such as overlapping appointments or double-bookings, to prevent scheduling inconsistencies.
- **Compliance Verification:** Validates schedules against regulatory requirements, organizational policies, and predefined standards to ensure compliance.

6.5 TIME TABLE GENERATOR

The Time Table Generator component automates the generation of time tables or schedules for various activities or events within the system. It includes functionalities such as:

- **Automated Scheduling:** Utilizes algorithms to automatically generate optimized schedules based on input parameters, constraints, and preferences.

These components collectively form the foundation of the timetable management system, providing robust capabilities for user access management, staff information management, resource allocation, schedule verification, and timetable generation.

CHAPTER 7

REQUIREMENT SPECIFICATION

7.1 INTRODUCTION

The system utilizes a constraint-based approach, integrating staff availability, course requirements, classroom capacities, and other constraints to generate optimal timetables. We explore various scheduling algorithms, including Constraint Satisfaction Programming (CSP), Genetic Algorithms (GA), and hybrid approaches, to find solutions that balance conflicting objectives and constraints effectively.

The implementation of the system involves a systematic approach encompassing data gathering, constraint definition, modeling, software development, testing, and deployment. Through the deployment of the system, colleges can expect to experience improved efficiency, reduced administrative burden, and enhanced satisfaction among staff and students.

In the subsequent sections of this paper, we delve into the details of our proposed system, outlining the methodology, algorithms, and implementation strategies employed. We also discuss potential enhancements and optimizations, such as machine learning techniques for adaptive scheduling and integration with student preferences, to further improve the effectiveness of the system.

Therefore, our system represents a significant advancement in the realm of college timetable scheduling, leveraging automation to streamline operations and enhance organizational effectiveness. By prioritizing the needs of teaching staff and embracing technological innovation, colleges can navigate the complexities of scheduling with greater ease and efficiency, ultimately enriching the academic experience for all stakeholders.

7.2 HARDWARE REQUIREMENTS

- **Processor :** Intel 15(8th Gen)
- **Ram :** 4GB
- **Internal Space:** 100GB

7.3 SOFTWARE SPECIFICATION

Software : pycharm, wampserver

Model: Genetic Algorithm

Languages:

- **Front end :** HTML, CSS, JS
- **Backend :** Flask python
- **Database :** MYSQL

7.4 GENETIC ALGORITHM

Time Table Scheduling problem is representative of the class of, NP-hard, combinatorial optimization problems with multi-constraints and for this we are using genetic algorithms (GA). GA has been found to be capable of getting optimized solutions for a number of problems like Time Table Scheduling for which no other algorithm exists. It is inspired by nature as altering a population of candidate solutions until a maximum or minimum solution is found. Scheduling a Timetabling faces two types of constraints, namely soft constraints and hard constraints. Soft constraints are those if we violate them in scheduling, the output is still valid, but hard constraints are those which if we violate them; the timetable is no longer valid. The search space of a timetabling issue is excessively tremendous, numerous arrangements exist in the search space and some of the arrangements are not possible. Possible arrangements here mean those which don't damage hard constraints and as well attempt to fulfill soft constraints. We have to pick the most suitable one from possible arrangements. Most suitable ones here mean those which don't disregard soft constraints to a more noteworthy degree. In this paper hard- constraints have been dealt with carefully and it has been guaranteed that soft constraints are too pursued however much as could be expected.

7.5 GENERAL WORKING OF GENETIC ALGORITHM

In a genetic algorithm, a population of chromosomes consisting of a given random collection of genes is initiated according to the following steps:

1. Generating an initial population of chromosomes.
2. Evaluating the suitability of each chromosome (individual) that forms the population.
3. Selecting the chromosomes for mating based on the above results.
4. Producing offspring by mating (cross over) the selected chromosomes.
5. Mutating genes randomly.
6. Repeating steps 3-5 until a new population is generated.
7. Ending the algorithm when the best solution obtained has not changed after a preset number of generations.

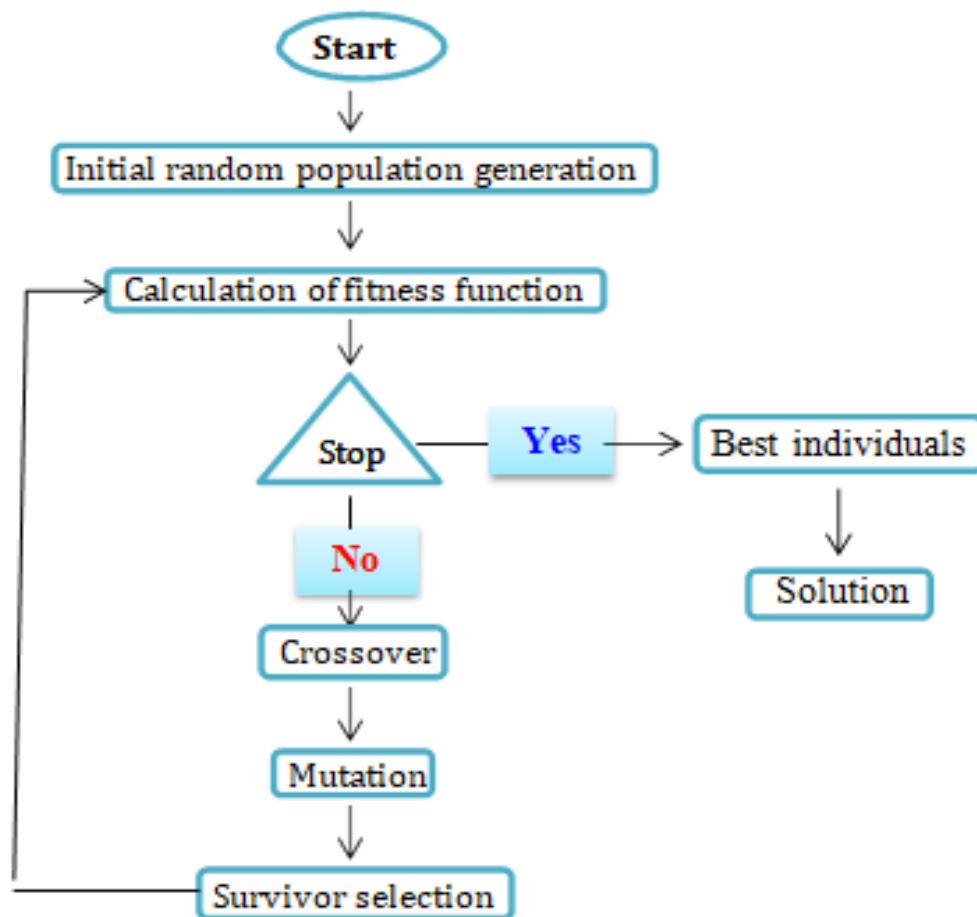


Fig 7.1 Flow Chart for Genetic Algorithm

7.6 WORKING OF GENETIC ALGORITHM IN TIME TABLE GENERATOR

Initialization

The Genetic Algorithm begins by initializing a population of potential solutions to the timetable scheduling problem. Each solution represents a possible timetable, with staff assignments to classes and rooms.

Fitness Evaluation

Once the initial population is generated, each solution is evaluated for its fitness or suitability. In the context of timetable scheduling, fitness evaluation involves assessing the quality of each timetable based on predefined criteria and constraints. This could include factors such as minimizing conflicts, maximizing staff preferences, and ensuring efficient resource utilization.

Selection

After fitness evaluation, a selection process is performed to choose the fittest individuals (timetables) from the population to proceed to the next generation. This selection process is typically based on the principle of "survival of the fittest," where individuals with higher fitness scores are more likely to be selected for reproduction.

Crossover

In the crossover stage, pairs of selected timetables are combined to create offspring or new solutions. This involves exchanging genetic information (class assignments, room allocations) between parent solutions to produce offspring solutions that inherit traits from both parents. The crossover process promotes diversity in the population and allows for the exploration of different combinations of assignments.

Mutation

After crossover, a mutation operation is applied to introduce random changes or variations into the offspring solutions. Mutation helps to maintain genetic diversity within the population

and prevents premature convergence to suboptimal solutions. In the context of timetable scheduling, mutation might involve randomly swapping staff assignments or adjusting class timings to explore new possibilities.

Fitness Evaluation II

Once the offspring population is generated through crossover and mutation, each new solution undergoes fitness evaluation to determine its suitability. This process repeats iteratively for multiple generations, with the aim of gradually improving the quality of solutions over time.

Termination

The Genetic Algorithm continues to iterate through selection, crossover, mutation, and fitness evaluation until a termination condition is met. This could be a predefined number of generations, a threshold fitness level, or a time limit. Once the termination condition is satisfied, the algorithm returns the best solution found as the final timetable scheduling solution.

CHAPTER 8

TESTING AND VALIDATION

The testing and validation phase of the project is critical for ensuring the reliability, functionality, and performance of the staff-driven automatic college timetable scheduling system. This section outlines the testing strategies, methodologies, and results obtained during the testing and validation process.

8.1 UNIT TESTING

Unit testing involves testing individual components or units of the system in isolation to ensure they function correctly. Test cases are designed to verify the behavior of each module, class, or function against expected outcomes. Unit tests are automated and executed frequently throughout the development process using testing frameworks such as Pytest or Django's built-in testing tools.

Unit Testing Result

Unit tests were conducted for each module and component of the system, covering functionalities such as user authentication, staff details management, constraint-based scheduling, and reporting. The majority of unit tests passed successfully, indicating that individual components function as expected and meet specified requirements.

8.2 INTEGRATION TESTING

Integration testing focuses on verifying the interactions and interfaces between different components of the system. Test cases are designed to validate the integration of modules, data flow between components, and communication between subsystems. Integration tests ensure that the system functions as a cohesive whole and that individual components work together.

Integration Testing Result

Integration tests were performed to verify the interactions between different modules and subsystems of the system. Integration tests confirmed that data flows correctly between

components, and that interfaces are implemented according to design specifications. A few integration issues were identified and addressed during testing, ensuring the overall integrity and coherence of the system.

8.3 USER ACCEPTANCE TESTING (UAT)

User acceptance testing involves validating the system against user requirements and expectations. Test cases are designed based on user stories, scenarios, and use cases, and are executed by end users or representatives from the target user group. UAT ensures that the system meets user needs, is intuitive to use, and delivers value to stakeholders.

User Acceptance Testing (UAT) Result

User acceptance tests were conducted with representatives from the target user group, including administrators, staff members, and students. Feedback from UAT sessions was generally positive, with users reporting satisfaction with the system's usability, functionality, and performance. Minor usability issues and feature requests were identified and addressed to improve user experience.

8.4 PERFORMANCE TESTING

Performance testing assesses the system's responsiveness, scalability, and stability under various load conditions. Test cases are designed to measure response times, throughput, resource utilization, and other performance metrics. Performance tests help identify bottlenecks, optimize system resources, and ensure that the system can handle expected workloads without degradation in performance.

Performance Testing Result

Performance tests were carried out to evaluate the system's responsiveness and scalability under simulated load conditions. The system demonstrated satisfactory performance, with response times within acceptable limits and resource utilization well within capacity thresholds. Performance tests provided confidence that the system can handle expected workloads effectively and efficiently.

CHAPTER 9

CODING AND SAMPLES OUPUTS

9.1 SOURCE CODE

```
import os

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from werkzeug.utils import secure_filename

import ar_master

from flask import Flask, render_template, flash, request, session, current_app,
send_from_directory, redirect, url_for

from datetime import date

import random

app = Flask(__name__, static_folder="static")

app.config.from_object(__name__)

app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'

mm = ar_master.master_flask_code()

@app.route("/")

def homepage():
```

```

return render_template('index.html')

@app.route("/admin", methods=['GET', 'POST'])

def admin():

    if request.method == 'POST':

        un = request.form['uname']

        pa = request.form['pass']

        usern = mm.select_direct_query("select * from admin_details where username='" + str(un) + "'
and password='" + str(pa) + "'")

        if usern:

            session['admin'] = un

            return redirect('/admin_home')

        else:

            return render_template('admin.html', msg='failed')

            return render_template('admin.html')

@app.route("/admin_home")

def admin_home():

    return render_template('admin_home.html')

@app.route("/admin_add_staff", methods = ['GET', 'POST'])

def admin_add_staff():

```

```

if request.method == 'POST':

    name = request.form['name']

    department = request.form['department']

    staff_id = request.form['staff_id']

    contact = request.form['contact']

    email = request.form['email']

    address = request.form['address']

    experience= request.form['experience']

    password= request.form['password']

    maxin=mm.find_max_id("staff_details")

    result=mm.insert_query("insertintostaff_details
values('"+str(maxin)+"','"+str(name)+"','"+str(department)+"','"+str(staff_id)+"','"+str(contact)+"'
','"+str(email)+"','"+str(address)+"','"+str(experience)+"','"+str(password)+"','0','0')")

    if (result == 1):

        return          render_template('admin_add_staff.html',flash_message=True,msg="Registration
Successfully")

    else:

        return render_template('admin_add_staff.html', msg="Failed")

    return render_template('admin_add_staff.html')

@app.route("/admin_view_staff",methods = ['GET', 'POST'])

```

```

def admin_view_staff():

data=mm.select_direct_query("select
id,name,department,staff_id,contact,email,address,experience from staff_details")

return render_template('admin_view_staff.html',items=data)

@app.route("/admin_view_student",methods = ['GET', 'POST'])

def admin_view_student():

data=mm.select_direct_query("select
id,roll_no,reg_no,name,contact,email,address,dob,gender,father_name,department,year,section
from student_details")

return render_template('admin_view_student.html',items=data)

import pymysql

class master_flask_code:

def __init__(self):

self.user = 'root'

self.password = ""

self.host = 'localhost'

self.database = 'python_ai_driven_staff_allocation'

def find_max_id(self,table):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
database=self.database,charset='utf8')

cursor = conn.cursor()

cursor.execute("SELECT id FROM "+table)

```

```

data = cursor.fetchall()

maxin = len(data)

if maxin == 0:

    maxin = 1

else:

    maxin += 1

return maxin

def insert_query(self,qry):

    conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
    database=self.database,charset='utf8')

    cursor = conn.cursor()

    result=cursor.execute(qry)

    conn.commit()

    conn.close()

    return result

def select_login(self,qry):

    conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
    database=self.database,charset='utf8')

    cursor = conn.cursor()

    cursor.execute(qry)

    data = cursor.fetchall()

    check = len(data)

```

```

if check == 0:

    return 'no'

else:

    return 'yes'

def select_single_column(self,table,column):

    conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
    database=self.database,charset='utf8')

    qry1=("select "+column+" from "+table)

    cursor = conn.cursor()

    cursor.execute(qry1)

    data = cursor.fetchall()

    return data

def select_direct_query(self,qry):

    conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
    database=self.database,charset='utf8')

    cursor = conn.cursor()

    cursor.execute(qry)

    data = cursor.fetchall()

    return data

```

9.2 SAMPLE SCREENSHOTS

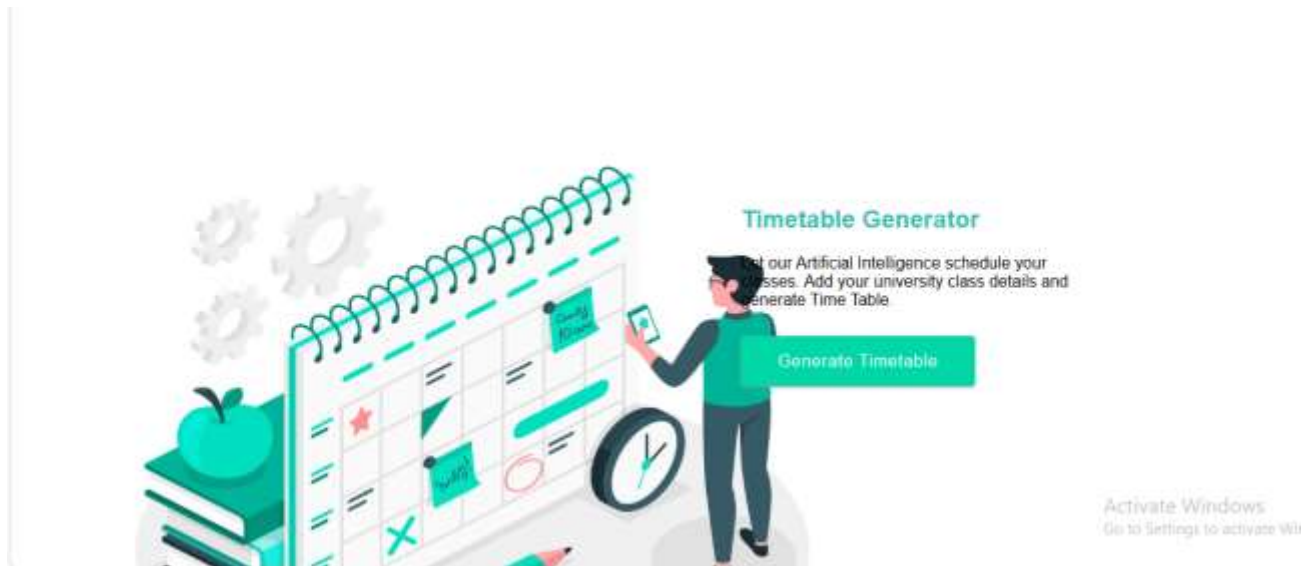


Fig: 9.1 Home page



Fig 9.2 Adding Staff Details

Add Course **Edit Courses**

Course number:

Course name:

Max numb students:

Instructors:

Submit

Activate Windows
Go to Settings to activate

FIG 9.3 Adding course details

Add Department **Edit Department**

Dept name:

Courses:

Submit

Activate Windows
Go to Settings to activate

Fig 9.4 Adding Department

Add Section **Edit Section**

Section id:

Department:

Num class in week:

Activate Windows
Go to Settings to activate Windows.

Fig 9.5 Adding Section

K18PV (Computer Science)

Class #	Course	Venue(Block- Room)	Instructor	Class Timing
0	C1 Java	14 503	T7 Goutam Mishra	T2 Tuesday 11:30 - 12:30
1	C1 Java	34 504	T6 Nandidni	Th2 Thursday 2:30 - 3:30
2	C1 Java	34 504	T6 Nandidni	T1 Tuesday 10:30 - 11:30
3	C1 Java	34 504	T6 Nandidni	T2 Tuesday 11:30 - 12:30
4	C1 Java	37 808	T7 Goutam Mishra	T1 Tuesday 10:30 - 11:30
5	C2 OS	16 208	T1 Abhik	Th2 Thursday 2:30 - 3:30
6	C2 OS	38 815	T3 Dr Isha	T4 Tuesday 2:30 - 3:30
7	C2 OS	13 101	T1 Abhik	T5 Tuesday 3:30 - 4:30
8	C2 OS	37 808	T3 Dr Isha	T5 Tuesday 3:30 - 4:30
9	C2 OS	57 705	T3 Dr Isha	M4 Monday 12:30 - 1:30

Fig 9.6 Final Time Table

CHAPTER 10

CONCLUSION AND FUTURE WORK

10.1 CONCLUSION

The automated college timetable scheduling system presented in this project represents a significant advancement in the field of academic scheduling. By leveraging automation, advanced algorithms, and user-driven preferences, the system aims to streamline the timetable generation process while improving efficiency and satisfaction among stakeholders.

Through the implementation of the system, several key benefits are expected to be realized. Administrators will experience reduced administrative burden and improved accuracy in timetable generation, leading to more efficient academic operations. Teaching staff will have greater flexibility in scheduling and increased satisfaction through consideration of their preferences. Students will benefit from more optimized timetables, leading to a better learning experience.

The system's architecture allows for scalability and adaptability to accommodate the evolving needs of educational institutions. Additionally, the use of modern software development practices such as modular design, automated testing, and continuous integration ensures the system's reliability and maintainability over time.

10.2 FUTURE ENHANCEMENT

While the current system provides a solid foundation for automated college timetable scheduling, there are several areas for future enhancement and expansion:

- **Real-Time Updates:** Implement real-time updating capabilities to allow for dynamic adjustments to schedules in response to changing circumstances or preferences.
- **Mobile Application:** Develop a mobile application to provide staff and administrators with convenient access to the system from their smartphones or tablets.

- **Intelligent Conflict Resolution:** Enhance the system's conflict resolution capabilities by incorporating intelligent algorithms that prioritize constraints and preferences effectively.
- **Integration with Student Systems:** Integrate the timetable scheduling system with student information systems (SIS) to ensure alignment between course offerings and student schedules.
- **User Feedback Mechanisms:** Implement features to gather feedback from users and stakeholders to continuously improve the system based on their needs and preferences.

By addressing these areas of future work, the automated college timetable scheduling system can continue to evolve and provide even greater value to educational institutions, staff, and students.

REFERENCES

- [1] Ahmed, K., & Keedwell, E. (2023). "A Hidden Markov Model Approach to the Problem of Heuristic Selection in Hyper Heuristics with a Case Study in High School Timetabling Problems," Massachusetts Institute of Technology, 2017.
- [2] Burke & Carter (2022). Practice and Theory of Automated Timetabling: Selected Papers from the 2nd International Conference. Lecture Notes in Computer Science, 14 (08), 115 129.
- [3] Esraa, A., & Ghada, E., K. (2022). "A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA)," Alexandria Engineering Journal, vol. 55, no. 2, pp. 1395-1409, 2016.
- [4] Fatih, C. & Merve, K. (2020). "A Fuzzy Logic and BinaryGoal Programming-Based Approach for Solving the Exam Timetabling Problem to Create a Balanced-Exam Schedule," International Journal of Fuzzy Systems, pp. 119-129.
- [5] Hamed, B., Jaber, K. & Amin, H. (2019). "Generating an optimal timetabling for multi-departments common lecturers using hybrid fuzzy and clustering algorithms," Methodologies and Application, vol. 23, p. 4735–4747.
- [6] Iosif, K., Ioannis, T. & Grigorios, B. (2023). "A Comparative Study of Modern Heuristics on the School," Algorithms, vol. 8, no. 3, pp. 723-742.
- [7] Joselynn, H., F. (2019). "The Effect of the Gainful Employment Regulatory Uncertainty on Student Enrollment at For Profit Institutions of Higher Education," Research in Higher Education, p. 1 25.
- [8] Kheiri, A., & Ed, K. (2018) "A Hidden Markov Model Approach to the Problem of Heuristic Selection in Hyper Heuristics with a Case Study in High School Timetabling Problems," Evolutionary Computation, vol. 25, no. 3, pp. 473- 501.
- [9] Mahmoud, S., Mahmoud, N., & Mohammad, M., M. (2020). "Optimal Localization of Shopping Centers Using Metaheuristic Genetic Algorithm," Journal of Optimization in Industrial Engineering, vol. 13, no. 1, pp. 167 176. [10] Nelishia, P., & Ender, O. (2020). "Automated

generation of constructive ordering heuristics for educational timetabling," *Annals of Operations Research*, vol. 275, no. 1, p. 181–208.

[10] K. Nguyen, D. Nguyen, K. Trieu, and N. Tran, "Automating a real-world university timetabling problem with Tabu search algorithm," in *2010 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, 2010

[11] A. Azlan and N. M. Hussin, "Implementing graph coloring heuristic in construction phase of curriculum-based course timetabling problem," in *2013 IEEE Symposium on Computers & Informatics (ISCI)*, 2013.

[12] R. E. Febrita and W. F. Mahmudy, "Modified genetic algorithm for high school time-table scheduling with fuzzy time window," in *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, 2017.

[13] T. Elsaka, "Autonomous generation of conflict-free examination timetable using constraint satisfaction modelling," in *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, 2017.

[14] F. D. Wihartiko, H. Wijayanti, and F. Virgantari, "Performance comparison of genetic algorithms and particle swarm optimization for model integer programming bus timetabling problem," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 332, p. 012020, 2018.

[15] A. Aminu et al., "Design and implementation of an automatic examination timetable generation and invigilation scheduling system using genetic algorithm," in *2nd International Conference on Applied Engineering (ICAE)*, 2019.

[16] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Comput.*, vol. 22, no. 2, pp. 387–408, 2018.

[17] K. Y. Junn, J. H. Obit, and R. Alfred, "The study of genetic algorithm approach to solving university course timetabling problem," in *Lecture Notes in Electrical Engineering*, Singapore: Springer Singapore, 2018, pp. 454–463.

- [18] E. Psarra and D. Apostolou, “Timetable scheduling using a hybrid particle swarm optimization with local search approach,” in 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA), 2019.
- [19] T. W. Ekanayake, P. Subasinghe, S. Ragel, A. Gamage, and S. Attanayaka, “Intelligent timetable scheduler: A comparison of genetic, graph coloring, heuristic and iterated local search algorithms,” in 2019 International Conference on Advancements in Computing (ICAC), 2019.
- [20] A. Schaerf, Local search techniques for large high-school timetabling problems, IEEE Transactions on Systems Man. And Cybernetics, Part A, 29 (4), 368- 377 (1999).
- [21] H.P. Hariyadi, T. Widiyaningtyas, et. al., Implementation of Genetic Algorithm to Academic Scheduling System, IEEE, Region 10 Conference, (2016).
- [22] S. Daskalaki, T. Birbas, E. Housos, An integer programming formulation for a case study in university timetabling, Eur. J. Oper. Res., 153, 117-135 (2004)
- [23] A.F. AbouElhamayed, A.S. Mahmodd, et. al., An Enhanced Genetic Algorithm-Based Timetabling System with Incremental Changes, IEEE, (2016).
- [24] M. Dimopoulou, P. Miliotis, An automated university course timetabling system developed in a distributed environment: A case study, Eur. J. Oper. Res., 153 136– 147 (2004).
- [25] E. Burke, K. Jackson, J. H. Kingston, R. Weare, Automated university timetabling: The state of the art, The Computer Journal, 40 (9): 565-571, (1997).
- [26] Rosen, Kenneth H. (2012). Discrete Mathematics and Its Applications (7th ed.). New York: McGraw-Hill. p. 119. ISBN 978-0-07-338309-5
- [27] D. Shiffman, The Nature of Code, Chapter 9, 2012, Creative Commons, 444 Castro Street, Suite 900, Mountain View, California 94041, USA. ISBN-13: 978- 0985930806.
- [28] E. R. Pianka, Evolutionary Ecology, 6th ed. San Francisco, CA: Addison-Wesley-Longman, 2000; Deepti Gupta, Shabina Ghafir, International Journal of Emerging Technology and Advanced Engineering, 2(5), (2012).

[29] Shraddha Thakare, Tejal Nikam, Prof, Mamta Patil, Automated Timetable Generation using Genetic Algorithm, International Journal of Engineering Research & Technology (IJERT) ISSN: 12278-0181, Vol. 9 Issue 07, July-2022.